**VED ANALYSIS**

**ABSTRACT:**

Effective inventory management is crucial for businesses to ensure smooth operations, minimize costs, and maximize customer satisfaction. One popular technique used in inventory management is the Vital, Essential, and Desirable (VED) analysis. VED analysis helps categorize inventory items based on their criticality and facilitates strategic decision-making regarding procurement, stock control, and resource allocation.

VED analysis categorizes inventory items as Vital, Essential, or Desirable based on their criticality. Vital items require continuous availability, while Essential items can tolerate slight delays, and Desirable items have a lower impact on operations. Implementing VED analysis involves inventory classification, prioritizing actions, and resource allocation. It helps allocate attention and resources to critical items, mitigate stockout risks, optimize inventory costs, and enhance operational efficiency. VED analysis aids in informed decision-making, efficient resource allocation, and streamlined procurement processes. By prioritizing resources and reducing stockouts, businesses can improve inventory management, optimize costs, and enhance customer satisfaction.

**STEPS:**

1. Create a Flask application.
2. Import the necessary libraries (os, pandas, numpy, matplotlib.pyplot, Flask, render\_template, request, redirect, url\_for, ARIMA).
3. Create an instance of the Flask application.
4. Define the routes:
5. Route 1: '/' - Renders the index.html template.
6. Route 2: '/upload' - Accepts GET and POST requests and performs the VED analysis.
7. Create the index.html template:
8. Include a form to upload a file.
9. Create the results.html template:
10. Display the VED analysis results in a table.
11. Display the VED chart.
12. Implement the VED analysis:
13. Load the uploaded data into a pandas DataFrame.
14. Preprocess the data (e.g., fill missing values).
15. Group the data by a specified column and aggregate the values.
16. Filter out rows with out-of-range values.
17. Fit an ARIMA model and forecast for each group.
18. Perform the VED analysis based on the forecasted values.
19. Generate the VED chart using matplotlib.
20. Save the VED chart and forecasted results.
21. Run the Flask application.

**CODE:**

1. **Python file:**

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from flask import Flask, render\_template, request, redirect, url\_for

from statsmodels.tsa.arima.model import ARIMA

app = Flask(\_name\_)

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/upload', methods=['GET' , 'POST'])

def upload():

# Check if a file was uploaded

if 'file' not in request.files:

return redirect(url\_for('index'))

file = request.files['file']

file\_path = os.path.join('uploads', file.filename)

file.save(file\_path)

# Perform VED analysis

ved\_results, chart\_path = perform\_ved\_analysis(file\_path)

# Render the results template with VED analysis results and chart path

returnrender\_template('results.html',ved\_results=ved\_results, chart\_path=chart\_path)

def perform\_ved\_analysis(file\_path):

# Step 1: Load your data into a pandas DataFrame

data = pd.read\_csv(file\_path)

# Step 2: Preprocess the data

data = data.fillna(method='ffill')

# Step 3: Group the data by 'Sub\_Category' and aggregate the 'no\_of\_ratings' column

demand\_data = data.groupby('Sub\_Category')['no\_of\_ratings'].sum().reset\_index()

# Step 4: Filter out rows with out-of-range integer values in 'no\_of\_ratings'

demand\_data['no\_of\_ratings'] = pd.to\_numeric(demand\_data['no\_of\_ratings'], errors='coerce')

demand\_data = demand\_data.dropna(subset=['no\_of\_ratings'])

demand\_data['no\_of\_ratings'] = demand\_data['no\_of\_ratings'].astype(int)

# Step 5: Fit an ARIMA model and forecast for each Sub\_Category

forecasts = {}

for sub\_category, sub\_data in data.groupby('Sub\_Category'):

sub\_data = sub\_data.copy()

sub\_data['no\_of\_ratings'] = pd.to\_numeric(sub\_data['no\_of\_ratings'], errors='coerce')

sub\_data = sub\_data.dropna(subset=['no\_of\_ratings'])

sub\_data['no\_of\_ratings'] = sub\_data['no\_of\_ratings'].astype(int)

sub\_data = sub\_data.loc[sub\_data['no\_of\_ratings'] >= 0] # Drop rows with negative ratings

if len(sub\_data) > 1:

sub\_data = sub\_data.reset\_index(drop=True) # Reset the index

model = ARIMA(sub\_data['no\_of\_ratings'], order=(1, 0, 1))

model\_fit = model.fit()

forecast = model\_fit.forecast(steps=1) # Forecast only one step ahead

if forecast is not None and len(forecast) > 0:

if isinstance(forecast, np.ndarray):

forecasts[sub\_category] = forecast[0]

elif isinstance(forecast, pd.Series):

forecasts[sub\_category] = forecast.iloc[0]

else:

forecasts[sub\_category] = np.nan

else:

forecasts[sub\_category] = np.nan

else:

forecasts[sub\_category] = np.nan

# Perform VED analysis

df\_demand\_forecast = pd.DataFrame({

'Sub\_Category': list(forecasts.keys()),

'Forecasted\_Demand': list(forecasts.values())

})

df\_demand\_forecast['Cumulative\_Demand'] = df\_demand\_forecast['Forecasted\_Demand'].cumsum()

total\_demand = df\_demand\_forecast['Forecasted\_Demand'].sum()

df\_demand\_forecast['Percentage\_Demand'] = df\_demand\_forecast['Forecasted\_Demand'] / total\_demand \* 100

df\_demand\_forecast['Cumulative\_Percentage'] = df\_demand\_forecast['Percentage\_Demand'].cumsum()

df\_demand\_forecast['VED\_Category'] = pd.cut(df\_demand\_forecast['Cumulative\_Percentage'],

bins=[0, 70, 90, 100],

labels=['Vital', 'Essential', 'Desirable'])

ved\_results = df\_demand\_forecast.to\_dict(orient='records')

# Count the number of Vital, Essential, and Desirable items

ved\_counts = df\_demand\_forecast['VED\_Category'].value\_counts()

# Generate the pie chart

ved\_labels = ['Vital', 'Essential', 'Desirable']

ved\_values = ved\_counts.values.tolist()

fig, ax = plt.subplots()

ax.pie(ved\_values, labels=ved\_labels, autopct='%1.1f%%', startangle=90)

ax.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle

chart\_path = os.path.join('static', 'ved\_chart3.png')

plt.savefig(chart\_path)

plt.close()

return ved\_results, chart\_path

if \_name\_ == '\_main\_':

app.run(debug=True)

1. **HTML, CSS, JAVASCRIPT :**

**Index.html:**

<!DOCTYPE html>

<html>

<head>

<title>File Upload</title>

<link href="/static/home.css" rel="stylesheet" />

</head>

<body>

<div class="home-container">

<div class="home-hero">

<div class="home-container1">

<div class="home-card">

<h1 class="home-text">&nbsp;&nbsp;&nbsp;VED ANALYSIS</h1>

<h2 class="home-text1">&nbsp;&nbsp;&nbsp;Upload and analyze your products</h2>

<div class="home-container2">

<div class="home-container3">

<div class="primary-pink-button-container">

<form action="{{ url\_for('upload') }}" method = "POST" enctype="multipart/form-data" >

<input type="file" name="file" id="fileInput" style="display: none;">

<button type="button" class="home-button button ButtonSmall" onclick="chooseFile()">Choose a File</button>

<!-- Add this to your index.html -->

<button class="home-button button ButtonSmall" type="submit">Upload</button>

</form>

</div>

</div>

</div>

</div>

</div>

</div>

<img alt="image" src="/static/curved6-1500h.jpg" class="home-image"/>

<img alt="image" src="/static/curved6-1500h.jpg" class="home-image-copy"/>

<section class="home-testimonials">

<img alt="image" src="/static/curved6-1500h.jpg" class="home-image1" />

</section>

</div>

<script>

function chooseFile() {

var fileInput = document.getElementById('fileInput');

fileInput.click(); }

</script>

</body>

</html>

**Results.html:**

<!DOCTYPE html>

<html>

<head>

<title>VED Analysis Results</title>

<style>

body {

background-color: white;

color: black;

padding: 20px;

}

.container {

text-align: center;

}

table {

border-collapse: collapse;

border: 1px solid black;

margin: 20px auto;

}

th, td {

border: 1px solid black;

padding: 10px;

}

img {

display: block;

margin: 20px auto;

}

h1, h2 {

color: black;

}

.gap {

margin-bottom: 40px;

}

</style>

</head>

<body>

<div class="container">

<h1>VED Analysis Results</h1>

<div class="gap"></div>

<h2>Results Table:</h2>

<table>

<tr>

<th>Sub Category</th>

<th>Forecasted Demand</th>

<th>Cumulative Demand</th>

<th>Percentage Demand</th>

<th>Cumulative Percentage</th>

<th>VED Category</th>

</tr>

{% for result in ved\_results %}

<tr>

<td>{{ result.Sub\_Category }}</td>

<td>{{ result.Forecasted\_Demand }}</td>

<td>{{ result.Cumulative\_Demand }}</td>

<td>{{ result.Percentage\_Demand }}</td>

<td>{{ result.Cumulative\_Percentage }}</td>

<td>{{ result.VED\_Category }}</td>

</tr>

{% endfor %}

</table>

<div class="gap"></div>

<h2>Chart:</h2>

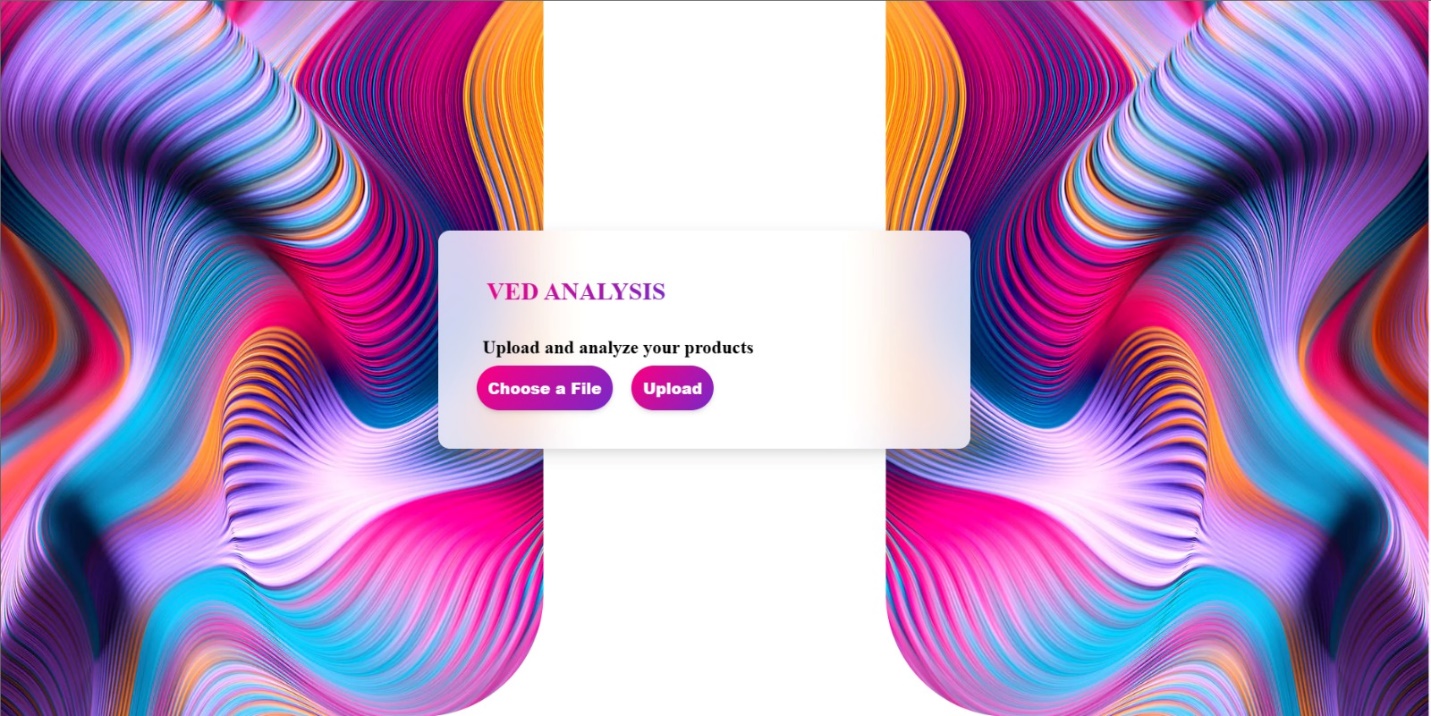
<img src="{{ url\_for('static', filename='ved\_chart3.png') }}" alt="VED Chart">

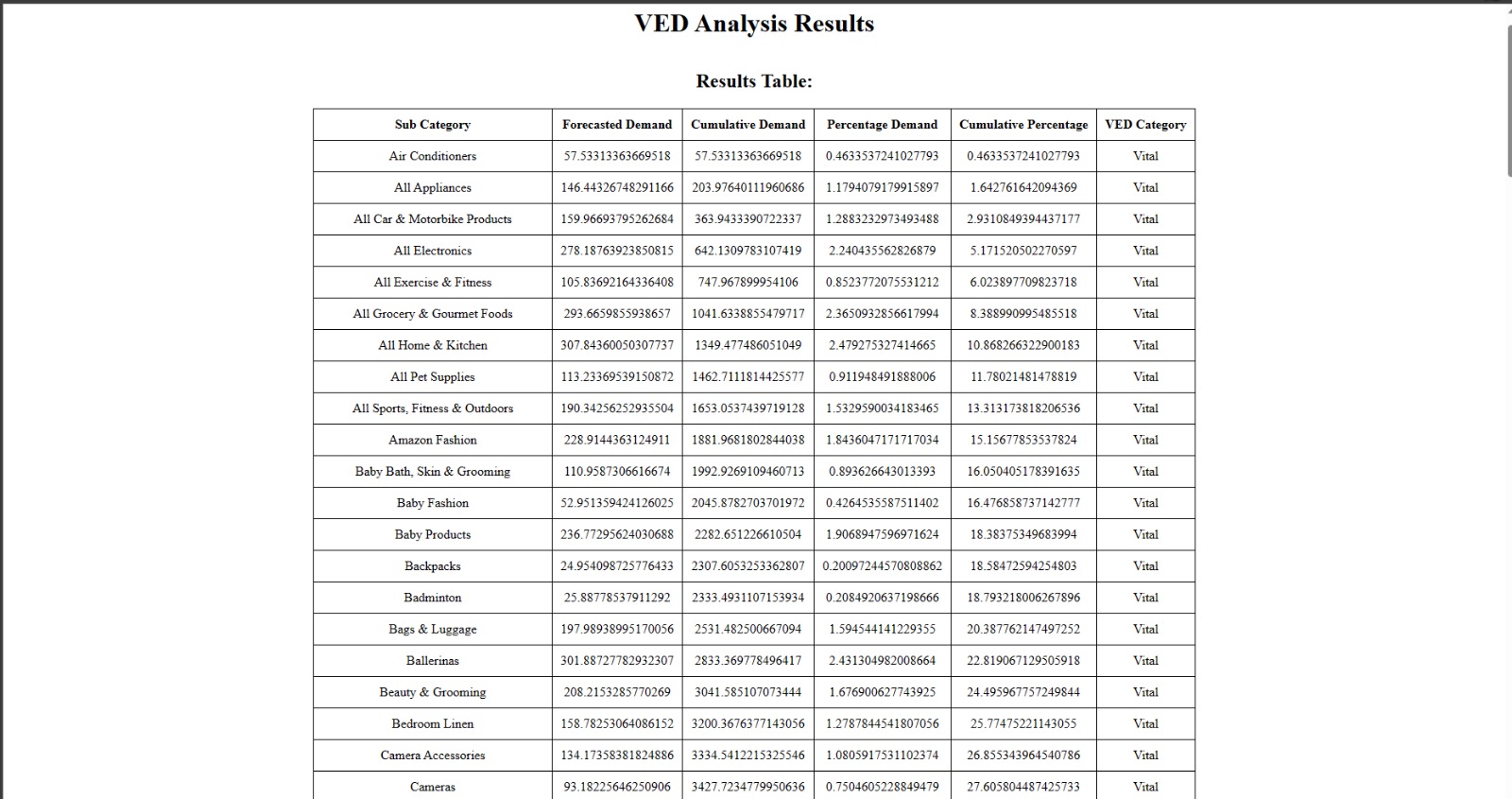
</div>

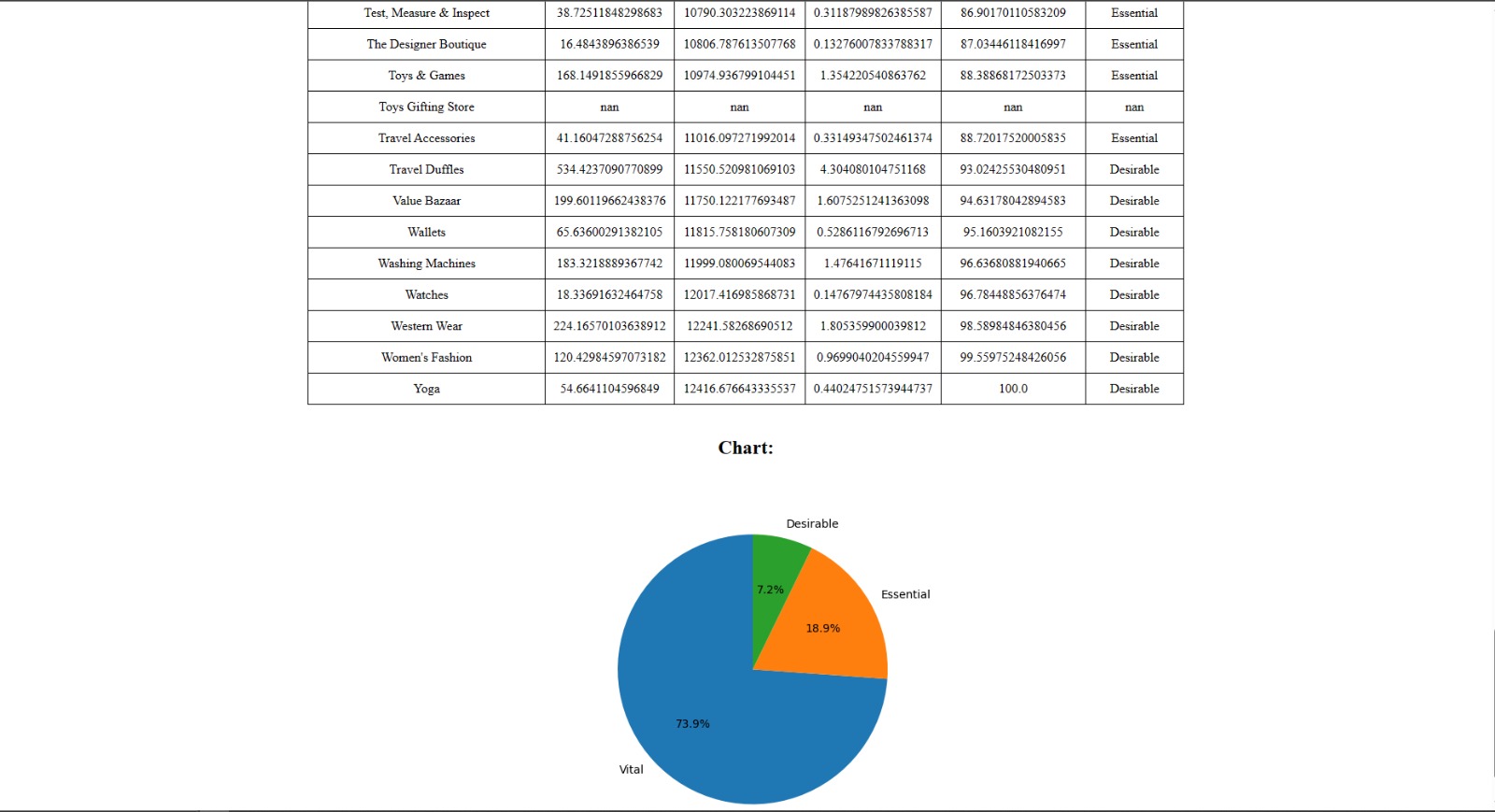
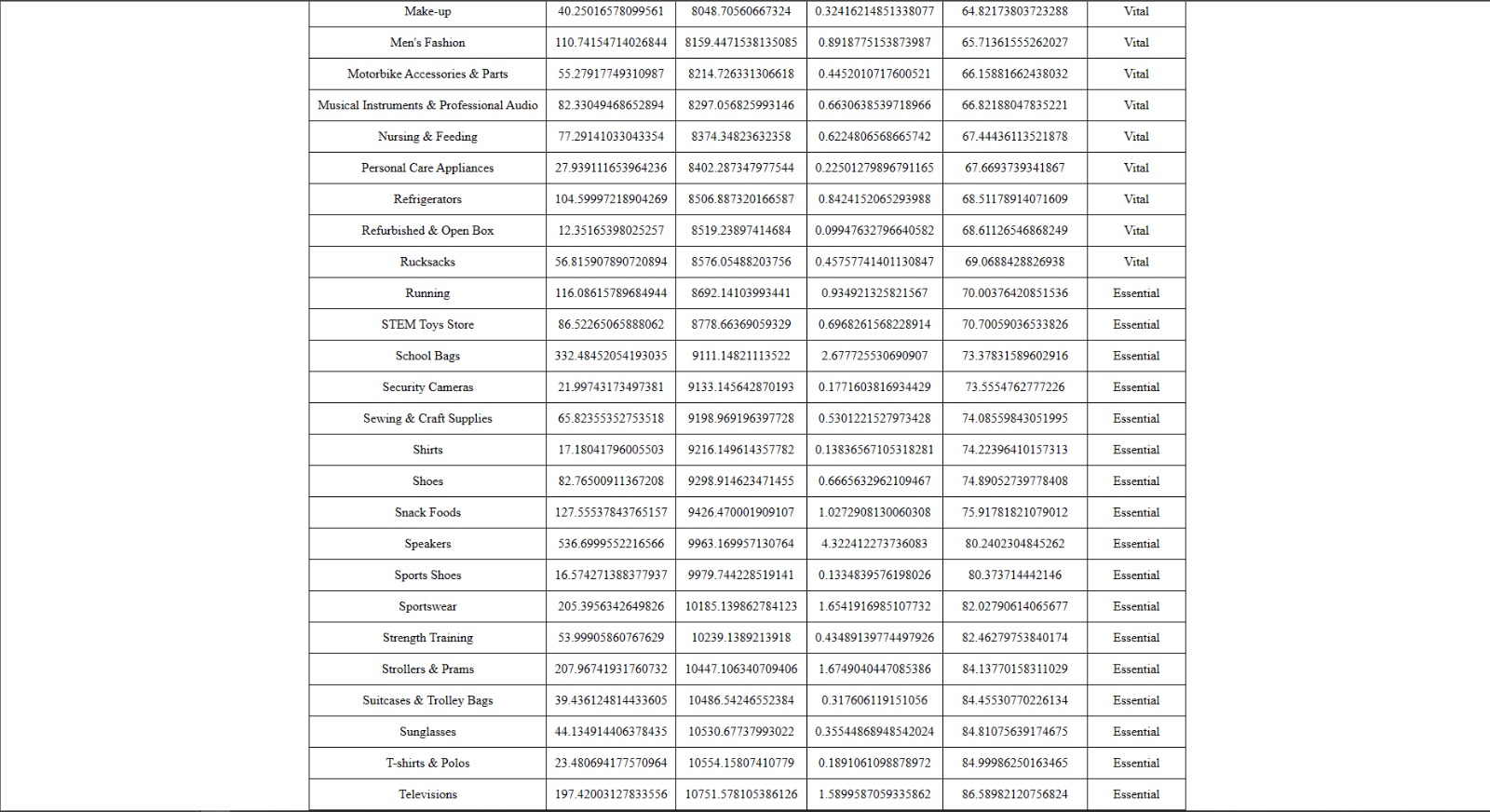
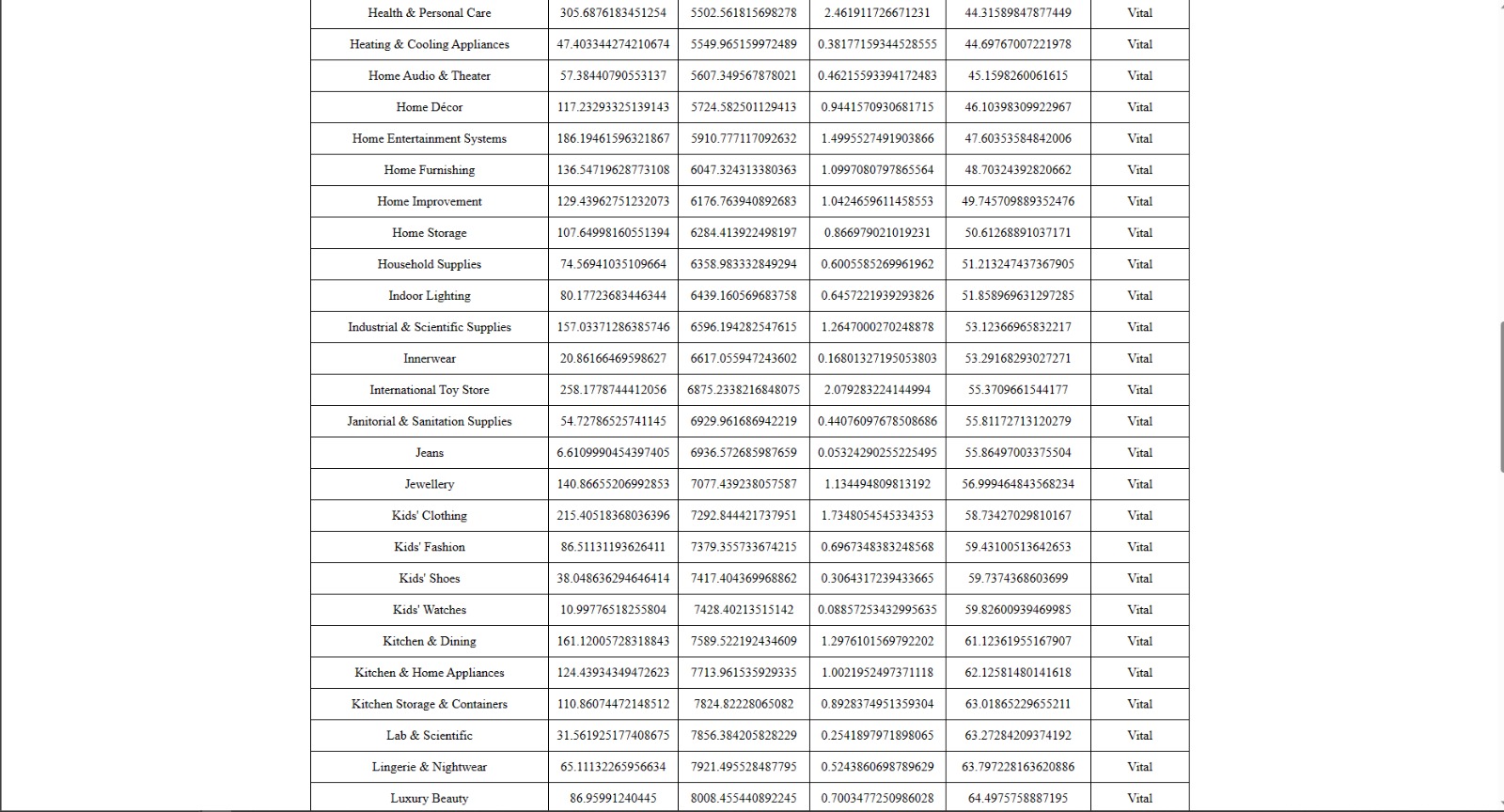
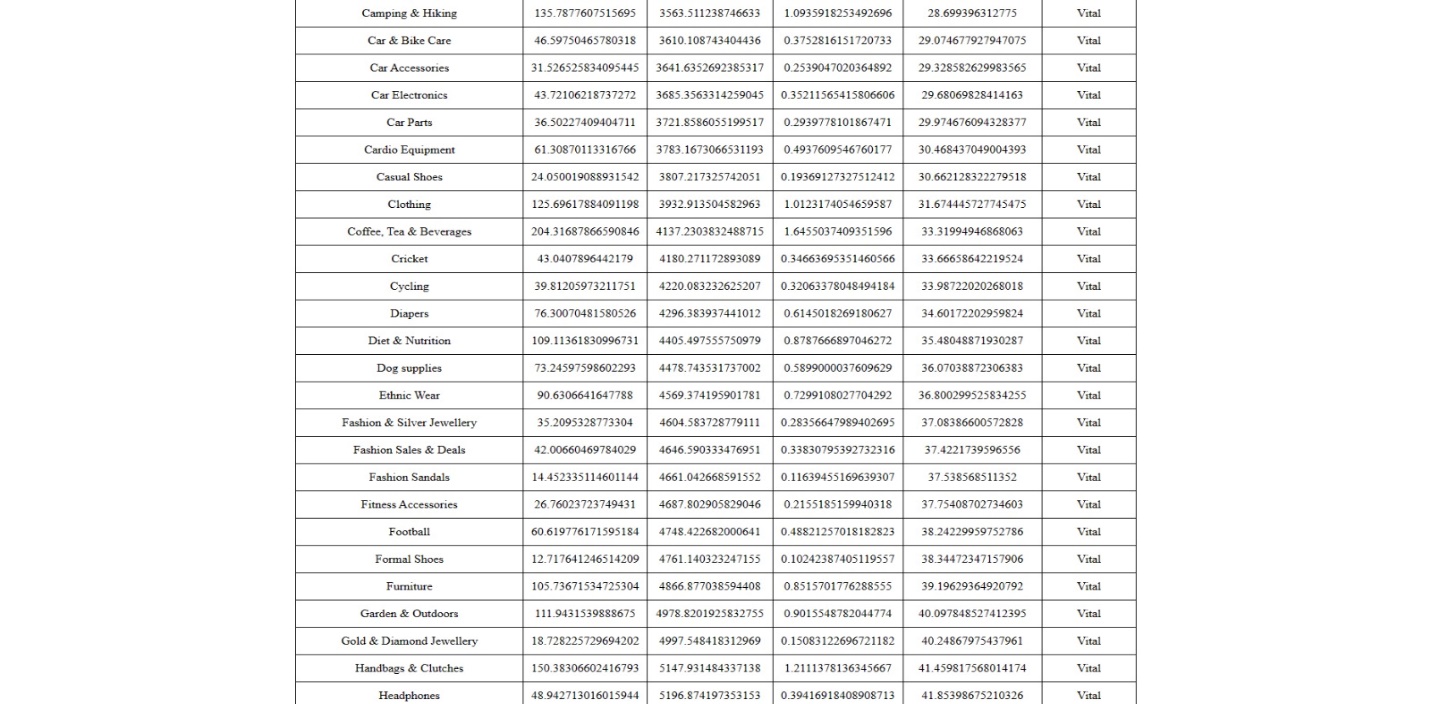
</body>

</html>

OUTPUT:







**INFERENCE:**

Based on the VED analysis results for Amazon product sales, the vital category accounts for the majority of forecasted demand at 73.9%. This indicates that a significant portion of the products analyzed are crucial for sustaining sales and should be closely monitored. The essential category represents 18.9% of the forecasted demand, indicating moderately important products that contribute to overall sales. The desirable category, with a share of 7.2%, consists of products with relatively lower demand. These findings can inform decisions related to inventory management, marketing strategies, and resource allocation to optimize sales performance on the Amazon platform.